**Escuela de Ingeniería**

**Departamento de Computación**

**Estructuras de datos y algoritmos fundamentales (TC1031)**

**Profesor: Dr. Leonardo Chang**

### Actividad práctica. Algoritmos de búsqueda

**ATENCIÓN: Subir este document con las respuestas en formato PDF!**

|  |  |
| --- | --- |
| **Título** | **Análisis de algoritmos de búsqueda** |
| **Aprendizaje esperado (objetivo)** | El alumno demostrará su capacidad para programar diferentes algoritmos de búsqueda y medir el tiempo de ejecución de los mismos bajo determinadas condiciones, así como analizar e interpretar los resultados obtenidos, comparando los diferentes algoritmos. |
| **Instrucciones** | Utilizando programación genérica (templates) y sobrecarga de operadores en C++, programa una clase Búsqueda que incluya los siguientes métodos y atributos:  Atributos:  vector<T> elementos;  Métodos:  Búsqueda secuencial  Búsqueda secuencial ordenada 1  Búsqueda secuencial ordenada 2  Búsqueda binaria  Cada método debe regresar el índice del elemento buscado, en caso de encontrarlo, ó -1 en caso de no encontrarlo.  Genere un arreglo de 100 000 números enteros de manera aleatoria.  Defina 5 números enteros y realice la búsqueda de cada número 10 veces con cada algoritmo.  Mida el tiempo de ejecución de cada caso y complete las tablas que aparecen más adelante en este documento.  Genere algunas gráficas (en Google Sheets) comparando los resultados de todos los algoritmos y sus tiempos de ejecución.  Analice e interprete los resultados alcanzados.  Realice una copia de este documento en Google Docs y complete las secciones indicadas más adelante.  Suba a la plataforma Canvas el archivo con sus resultados.  Suba a Github todos los códigos programados.  No se aceptan trabajos fuera de fecha ni por correo electrónico. |
| **Lugar en que se llevará a cabo** | Casa |
| **Forma de trabajo** | Individual |
| **Recursos** | Foros de información en Internet  Wikipedia (<http://www.wikipedia.org>)  Códigos de algoritmos vistos en la materia  Computadora |
| **Tiempo estimado** | 5 horas |

# Respuestas

Repositorio de GitHub:

[*https://github.com/DemiurgeApeiron/PDEDD/tree/master/AlogritmosDeBusqueda*](https://github.com/DemiurgeApeiron/PDEDD/tree/master/AlogritmosDeBusqueda)

*Excel*

[*https://tecmx-my.sharepoint.com/:x:/g/personal/a01028325\_itesm\_mx/ER7AGGQeX3BJn1NIeo3ttvsBXP6TSIXgWWFVqxBBvlSnNQ?e=sQ2ONs*](https://tecmx-my.sharepoint.com/:x:/g/personal/a01028325_itesm_mx/ER7AGGQeX3BJn1NIeo3ttvsBXP6TSIXgWWFVqxBBvlSnNQ?e=sQ2ONs)

Tablas con los resultados de las mediciones:

*[Completar las tablas que aparecen a continuación]*

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Búsqueda secuencial | | | | | | | | | | |
| **Tabla de resultados a completar (***tiempo en s***)** | | | | | | | | | | |
| **No.** | **B1** | **B2** | **B3** | **B4** | **B5** | **B6** | **B7** | **B8** | **B9** | **B10** |
| **60420** | 0.000860202 | 0.00125917 | 0.00065104 | 0.000676334 | 0.00103166 | 0.000634849 | 0.00325153 | 0.00144886 | 0.000986584 | 0.00109805 |
| 42000 | 0.000549731 | 0.00099148 | 0.00106683 | 0.000659713 | 0.000518266 | 0.00102554 | 0.00108579 | 0.00180337 | 0.00105352 | 0.00117954 |
| 19999 | 0.000701445 | 0.00114499 | 0.00062558 | 0.00107696 | 0.000605865 | 0.000983361 | 0.000587891 | 0.00115802 | 0.00204423 | 0.000480742 |
| 98278 | 0.00167978 | 0.0014857 | 0.000445198 | 0.00112891 | 0.000993575 | 0.000394176 | 0.00101152 | 0.000977368 | 0.000731834 | 0.00175047 |
| 10000 | 0.00105067 | 0.000674276 | 0.000970273 | 0.000744905 | 0.00134795 | 0.000851779 | 0.000990584 | 0.000561957 | 0.00106748 | 0.000896095 |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Búsqueda secuencial ordenada 1 | | | | | | | | | | |
| **Tabla de resultados a completar (***tiempo en s***)** | | | | | | | | | | |
| **No.** | **B1** | **B2** | **B3** | **B4** | **B5** | **B6** | **B7** | **B8** | **B9** | **B10** |
| **60420** | 0.000607561 | 0.000644648 | 0.000634337 | 0.000819411 | 0.000643968 | 0.000750469 | 0.000604673 | 0.000594977 | 0.000596973 | 0.000918285 |
| 42000 | 0.000483892 | 0.000559062 | 0.000467144 | 0.000502911 | 0.000582989 | 0.000803442 | 0.000900709 | 0.000465139 | 0.000432568 | 0.000469039 |
| 19999 | 0.000575495 | 0.00047211 | 0.000302767 | 0.000297336 | 0.00356028 | 0.000219018 | 0.000259634 | 0.000259926 | 0.000261003 | 0.00026078 |
| 98278 | 0.000411588 | 0.000265725 | 0.000997484 | 0.000951433 | 0.000956197 | 0.000966398 | 0.00100185 | 0.00100458 | 0.0011417 | 0.00119315 |
| 10000 | 0.000227086 | 0.000212109 | 0.000189528 | 0.000158983 | 0.00020391 | 0.00016102 | 0.000162327 | 0.000182826 | 0.000161661 | 0.000164423 |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Búsqueda secuencial ordenada 2 | | | | | | | | | | |
| **Tabla de resultados a completar (***tiempo en s***)** | | | | | | | | | | |
| **No.** | **B1** | **B2** | **B3** | **B4** | **B5** | **B6** | **B7** | **B8** | **B9** | **B10** |
| **60420** | 0.000346388 | 0.000324306 | 0.000361547 | 0.000512107 | 0.000348744 | 0.000331177 | 0.000327361 | 0.000411798 | 0.000324219 | 0.000396588 |
| 42000 | 0.00024444 | 0.000234854 | 0.000479237 | 0.000219118 | 0.000219213 | 0.000315625 | 0.000548994 | 0.000368858 | 0.000258555 | 0.000218072 |
| 19999 | 0.000311915 | 0.00022117 | 0.00011052 | 0.000117629 | 0.00361785 | 0.000107722 | 0.000106283 | 0.000106301 | 0.000108899 | 0.000108296 |
| 98278 | 0.000252237 | 0.000131964 | 0.000508559 | 0.000525317 | 0.000573818 | 0.000623904 | 0.000506237 | 0.000514365 | 0.000808831 | 0.000789245 |
| 10000 | 9.4221e-05 | 5.7882e-05 | 5.4852e-05 | 5.3891e-05 | 0.000756424 | 5.4551e-05 | 7.9121e-05 | 6.4467e-05 | 5.5755e-05 | 5.4355e-05 |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Búsqueda binaria | | | | | | | | | | |
| **Tabla de resultados a completar (***tiempo en s***)** | | | | | | | | | | |
| **No.** | **B1** | **B2** | **B3** | **B4** | **B5** | **B6** | **B7** | **B8** | **B9** | **B10** |
| **60420** | 3.826e-06 | 4.539e-06 | 4.958e-06 | 5.238e-06 | 3.421e-06 | 3.349e-06 | 3.36e-06 | 3.237e-06 | 3.043e-06 | 3.463e-06 |
| 42000 | 4.868e-06 | 3.553e-06 | 7.712e-06 | 2.868e-06 | 3.12e-06 | 4.738e-06 | 5.786e-06 | 3.583e-06 | 3.269e-06 | 3.168e-06 |
| 19999 | 5.004e-06 | 3.269e-06 | 3.149e-06 | 3.412e-06 | 0.0001546 | 2.978e-06 | 3.136e-06 | 3.371e-06 | 3.432e-06 | 3.169e-06 |
| 98278 | 5.189e-06 | 3.456e-06 | 2.842e-06 | 3.12e-06 | 3.303e-06 | 3.4438e-05 | 3.16e-06 | 3.243e-06 | 5.491e-06 | 5.312e-06 |
| 10000 | 7.332e-06 | 3.27e-06 | 3.248e-06 | 3.249e-06 | 5.671e-06 | 3.116e-06 | 3.4e-06 | 3.427e-06 | 3.058e-06 | 3.073e-06 |

Interpretación de los resultados:

*Al observar los resultados uno puede percatarse de cómo hay una gran diferencia entre los algoritmos, Por ejemplo es bastante visible que el algoritmo secuencial fue el más lento de todos con un promedio de 0.001019913s esto es debido a qué tiene una complejidad de O(n), ya que este algoritmo en el peor de los casos recorre toda la lista. En tercer lugar queda el algoritmo secuencial ordenado con un promedio de 0.00059329s, ya que este ocupa una lista ordenada y por lo tanto si el número por el que está pasando es mayor al número que está buscando detiene el algoritmo, sin embargo este algoritmo también tiene una complejidad de O(n) debido a que en el peor de los casos también va a recorrer toda la lista. En segundo lugar queda el algoritmo secuencial ordenado 2 con un promedio de 0.000365356s, ya que en este pasa por la lista con un incremento de 2 lo que significa en el peor de los casos va a tener una complejidad temporal O(n/2). En primer lugar queda el algoritmo más rápido el cuál es la búsqueda binaria con un promedio de 7.50034E-06s esto es debido aquí este algoritmo corta la lista a la mitad por cada secuencia que tiene y dependiendo de si el punto en el que se encuentra es mayor o menor al resultado que se está buscando va a escoger ir a la izquierda la derecha, cómo corta el input a la mitad cada vez que se ejecuta funciona como un exponencial invertida o un logaritmo, por lo cual su complejidad temporal es de O(log(n)) la cual si se busca en una tabla de complejidades temporales es una de las menores complejidades que hay y por lo cual es el algoritmo más rápido en este análisis.*

Gráficas comparativas